2

ARMTRAK: A Domain for the Unified Study of Natural Language, Planning, and Active Vision

N.G. Martin, J.F. Allen, and C.M. Brown

Technical Report 324
January 1990

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 324	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* ARMTRAK: A Domain for the Unified Study of Natural Language, Planning, and Active Vision		5. TYPE OF REPORT & PERIOD COVERED technical report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) N.G. Martin, J.F. Allen, and C.M. Brown		8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0193
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Dept. University of Rochester Rochester, NY 14627		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209-2308		12. REPORT DATE January 1990
		13. NUMBER OF PAGES 43 pages
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Office of Naval Research Computer Sciences Division Arlington, VA 22217-5000		15. SECURITY CLASS. *(of this report)* unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT *(of this Report)* Distribution of this document is unlimited		
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*		
18. SUPPLEMENTARY NOTES none		
19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)* natural language, planning, active vision		
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)* (over)		

20. ABSTRACT

ARMTRAK is a micro-world, based on the control of model trains, designed to integrate work in natural language, planning, vision and robotics. The primary advantage of the domain is that it provides examples that involve only few objects but that require sophisticated analysis. Because the examples involve few objects, the complex reasoning required is not intractable. On the other hand, more objects can be introduced to study techniques for tractable reasoning. Simple and complex examples in the same domain allow work at different levels to take place simultaneously. As a planning domain, ARMTRAK allows exercising planners in a real-time domain about which the planner has only imperfect knowledge. As a domain for natural language research, it allows research into the grounding of language in real situations, and the problem of coordinating the behavior of agents through language. As a domain for active vision research, it is challenging because it requires extracting information whose parameters cannot be completely specified beforehand. Two implementations of ARMTRAK have been developed: a simulation and a version using the Rochester Robot. The simulation allows work on real-time planning, and a robot version shows the feasibility of a real working system based on model trains.

ARMTRAK: A Domain for the Unified Study of Natural Language, Planning, and Active Vision

Nathaniel G. Martin, James F. Allen and Christopher M. Brown

January 31, 1990

Abstract

ARMTRAK is a micro-world, based on the control of model trains, designed to integrate work in natural language, planning, vision and robotics. The primary advantage of the domain is it provides examples that involve only few objects but that require sophisticated analysis. Because the examples involve few objects, the complex reasoning required is not intractable. On the other hand, more objects can be introduced to study techniques for tractable reasoning. Simple and complex examples in the same domain allow work at different levels to take place simultaneouslv. As a planning domain, ARMTRAK allows exercising planners in a real time domain about which the planner has only imperfect knowledge. As a domain for natural language research, it allows research into the grounding of language in real situations, and the problem of coordinating the behavior of agents through language. As a domain for active vision research, it is challenging because it requires extracting information whose parameters cannot be completely specified beforehand. Two implementations of ARMTRAK have been developed: a simulation and a version using the Rochester Robot. The simulation allows work on real time planning, and a robot version shows the feasibility of a real working system based on model trains.

Contents

List of Figures

1 Introduction

ARMTRAK is a micro-world to support a long-term research effort in integrating planning, natural language, vision and robotics under development at the University of Rochester. It is based on the control and monitoring of a model train set capturing the conceptual simplicity of blocks-world domains while extending it enough to exercise intelligent systems on more realistic problems. ARMTRAK consists of a sensorium that provides information about the current state of the world, and a set of commands that change the state of this world (see Figure 1). Both sensation and action are subject to failure, and costs are associated with gathering information or acting. Moreover, the information available through the sensorium is limited.

A computer simulation of this micro-world has been implemented, and we are currently building a version that will use the Rochester Robot [14] as the sensorium. This new "real" version of ARMTRAK will consist of the train set, a controller providing the interface between the computer and the train set, video cameras with appropriate supporting hardware and software for the vision capabilities, and a terminal acting as the natural language interface to a person.

ARMTRAK allows the testing of models of real-time plan-directed control, multi-agent planning, reasoning about simultaneous action, plan execution monitoring and debugging, complex visual processing such as object recognition, tracking moving objects, general reasoning about causality to predict future events (both undesirable ones such as trains de-railing or colliding and desirable one such as discovering an available car near where it is needed), and complex real-time dialogues involving the negotiation, discussion and formulation of plans, and cooperative man-machine plan execution.

Though it was initially proposed as a planning domain, implementation of ARMTRAK using the Rochester Robot and a set of model trains offers challenges for research in natural language. Because the domain can be restricted to a few objects, strategies for natural language understanding and generation can be investigated without encoding all human knowledge. At the same time, more complex interactions are possible because activity taking place may have unexpected consequences for both the system and the person communicating with it, and because the system may have goals and desires of its own.

While simple ARMTRAK configurations can be used to support initial

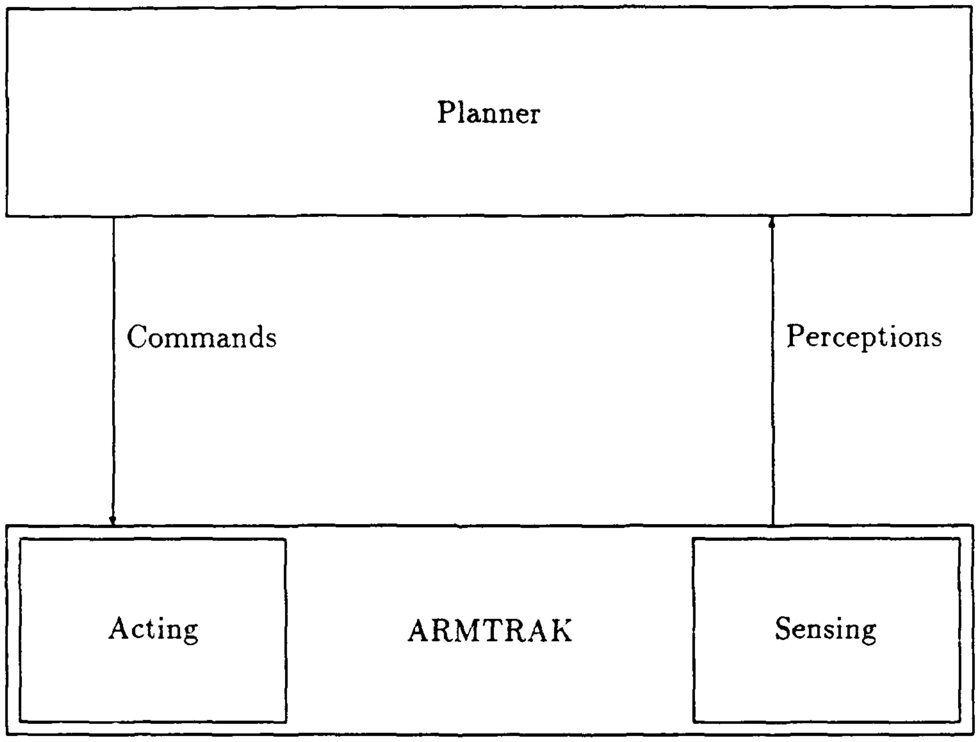

Figure 1: Architecture of ARMTRAK

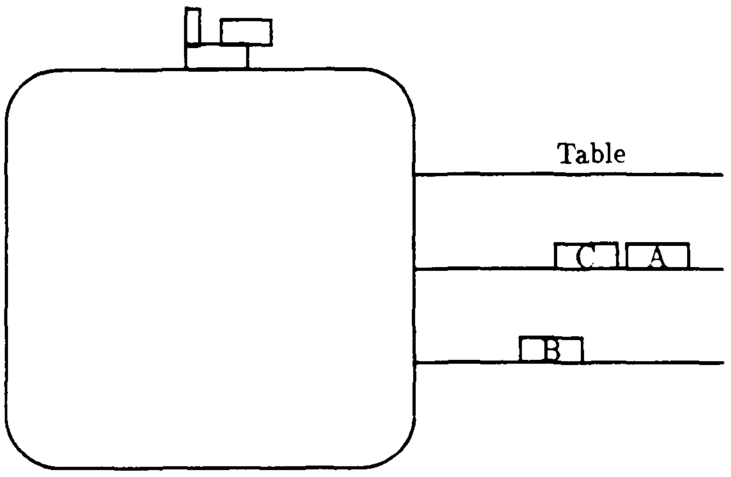

Figure 2: Sussman Anomaly in ARMTRAK

research, it can be easily extended to provide much more complex problems of perception and reasoning. A simple ARMTRAK domain, for instance, might include a circular track with several spurs, a single engine, two or three train cars, automatic switches and a train-car decoupler. With this setup, all the traditional blocks-world problems can be represented as train-car arrangement problems. The Sussman anomaly, as it would be represented in ARMTRAK, is shown in Figure 2.

To solve even such simple problems in ARMTRAK, however, the system needs to be able to perceive the position of the cars, monitor the engine as it picks up cars and drops them off at the right place, and use real-time "reactive" control mechanisms to control the engine. From here, the domain can become even more complex: the track may be extended requiring significant route planning; the number of cars may be increased to complicate perception and reasoning; more than one engine may be used to introduce problems of simultaneous action; a person may be introduced to interact with the system to negotiate goals; formulate plans and cooperatively aid in execution; a wall may be placed hiding part of the track so that the system cannot directly perceive the complete world; and so on.

It should be stressed that these are all domain-independent problems that arise whenever real-time control or man-machine dialogue is necessary.

ARMTRAK is used as a testbed for demonstrating the effectiveness of the particular theories. ARMTRAK is not a goal in itself and domain-specific solutions will not be used, except possibly in areas that are not the focus of the research.

Three planning issues arise in ARMTRAK. These are: dealing with time, dealing with uncertainty, and choosing appropriate goals. These issues will arise whenever autonomous agents deal with realistic problems. Consider, for example, the problem of scheduling ambulances. Time is crucial both because the patient must arrive at the hospital as soon as possible and because events occurring during the generation and the execution of plans are ordered by time. Events such as traffic and stop lights cannot be known in advance, so the information the planner deals with is uncertain. Goals must continually be weighed to make rational choices, say between risking an accident by driving too fast or risking the health of the patient by driving too slow. Similar problems will be faced by exploratory robots. The Mars lander will have a limited amount of time for exploration and should make the most of it. Moreover, it will be in an unknown and uncertain environment. It will also need to choose between performing the activities requested by its operators on Earth, and dealing with contingencies of which, due to the time lag in communication, only it is aware. Even simple warehouse robots face these problems. They must retrieve items as quickly as possible and must optimize their routes to do so. A large warehouse with a changing inventory will make it impossible to keep the robot's knowledge completely consistent with the contents of the warehouse. Such robots also need to prioritize among requests from multiple sources.

In this technical report we explore the three main problem areas to be investigated using ARMTRAK: planning, natural language and active vision. The report is divided into six sections. The next section provides an overview of the ARMTRAK system and an extended example that will be used throughout to suggest experiments than can be performed in ARMTRAK. The third, fourth and fifth sections discuss issues the ARMTRAK domain raises for planning, natural language and active vision respectively. The last section describes the state of ARMTRAK as of this writing.

Name	*Arguments*	
Set_Switch	Switch No.	New State
Set_Power	Loc. No.	Power Inc.
Uncouple	Track No.	

ARMTRAK commands

Name	*Arguments*	*Returns*
Get_Switch	Switch No.	State
Get_Speed	Train No.	Speed
Get_Locate	Train No.	Location

ARMTRAK queries

Figure 3: ARMTRAK queries and commands

2 ARMTRAK

2.1 Technical Description

The train set being used in the Robot implementation of ARMTRAK is an ordinary HO gauge model train set. Intuitions about operating such trains can therefore be freely applied to the ARMTRAK domain. A more formal description of the domain follows.

ARMTRAK consists of model train layout coupled with a sensorium. A set of commands manipulates the layout; the sensorium satisfies a set of queries. The layout consists of track and trains. The track comes curved or straight sections. The curved sections are categorized by the radius of the circle that the track describes. Switches are sections of track that bifurcate into two branches one curved and one straight. The state of the switch determines which of the two alternatives is taken. Decouplers are sections of track that can uncouple cars. To uncouple cars the engine pushes the coupled cars onto the decoupler, the decoupler is activated, and the train moves forward. Cars on the decoupler will not reconnect as long as the decoupler is active. The track is a sequence of track sections. Trains are composed of a locomotive and cars. Signals sent to the locomotive specify the power supplied to the engine. The velocity of the train is determined

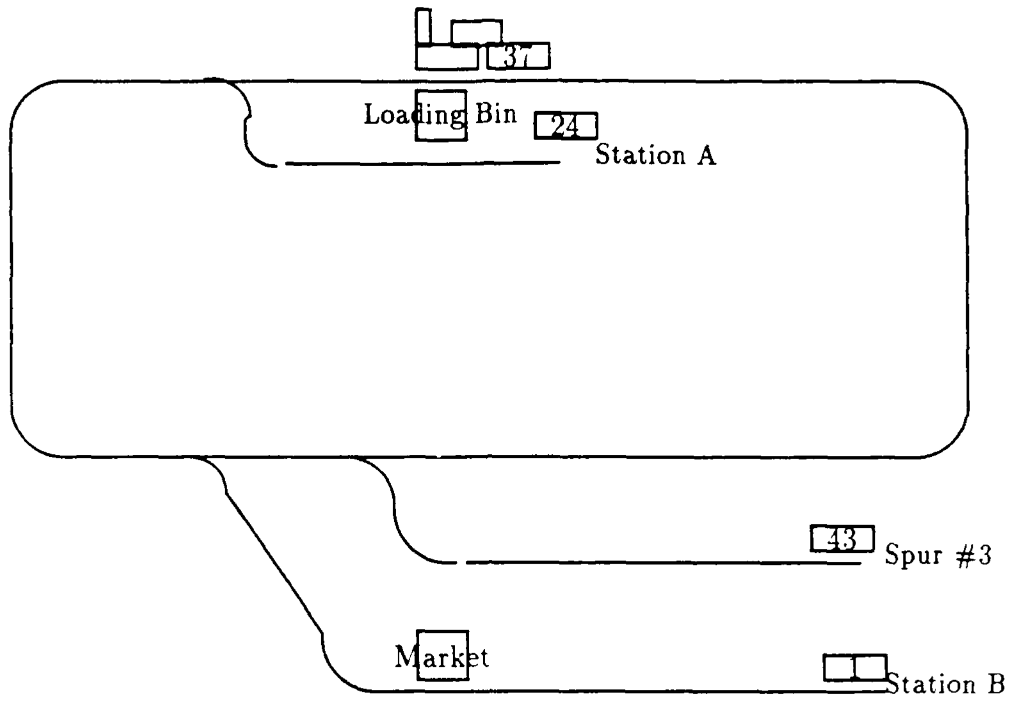

Figure 4: Example layout

by the power supplied to it, its weight, and the grade of the track it is on. When a car backs into another car the cars connect or couple. Commands can be sent to the switches, the trains and the decouplers. Commands to the switches change the state of the switch; commands to the engines change the engine's power; commands to the decouplers activate or deactivate the decoupler.

The architecture of the ARMTRAK planning domain is shown in figure 1. The planner sends commands to ARMTRAK that change the state of the world and receives perceptions back in response to queries. The commands and queries are shown in figure 3.

2.2 Example

To make the discussion of ARMTRAK as a testbed for intelligent planning and natural language processing more concrete, consider the following scenario. The system can control the trains, the switches on the tracks, and

System's Goal Leave cattle car 37 at station B

Person's Goals 1. Move a load of potatoes from the loading bin at Station A to the market and maintain the engine.

Initial State
- Engine has cattle car 37 attached.
- Box car 43 is on Spur #3.
- Cattle car 24 is at Station A.
- There are potatoes in the loading bin near Station A.

Figure 5: Example goals of person and system and initial situation

other devices such as the decoupler. It can also perceive most of the world at any time. Certain parts of the layout may, however, be visible only to the person. Furthermore, certain actions in the domain, say, loading cars, can only be done by the person. As one can see, this presents an excellent opportunity for plans that necessarily involve the person and the machine, and allows plans that cannot be executed without significant communication occurring between the man and the machine. A drawing on the layout on which the example is based appears in Figure 4.

There are two agents involved in this scenario: a person, P, and the system, S. Initially, the system's goal is to move the cattle car attached to the engine to Station B. The person's primary goals are to move some potatoes from Station A to the market near station B and to have then engine maintained. In addition, the person has a secondary goal of getting cattle car 24 to Station B.

The system's relation to the person is like that of an employee to a boss. The system may have goals of its own, and may choose to delay satisfying the person's goals in order to satisfy its own first. Still, in general, the person's goals are given higher priority than its own.[1]

[1]Thanks to David Traum for pointing out the importance of the issue of social relationship in a multi-agent domain.

3 Planning

Traditional planning has concentrated on issues of correctness and performance in controlled environments. Planning of this type has been well-served by the blocks world. This planning domain does not, however, address several new problems raised by attempts to build robots that plan actions in the real world. While planning domains must be simple enough for humans to reason about, they must also be complex enough to exercise planning programs. Though the blocks world admirably captures the first requirement, it fails on the second. The blocks world makes five simplifying assumptions that affect the planners that operate in it. These assumptions are:

1. only the planner has effects on the world,
2. the planner's actions always cause expected effects,
3. only one action can occur at one time,
4. a goal must be completely achieved for a plan to succeed and
5. complete information is always available.

Due to these simplifying assumptions, the blocks world fails to support sufficient complexity to exercise planning under three major categories of constraints:

1. real-time planning,
2. planning in an uncertain world, and
3. choosing among multiple, possibly contradictory, goals.

ARMTRAK will have advantages over the blocks world due to increased flexibility along four axes. First, time in ARMTRAK is more like time in the common sense world. This increased realism allows experiments involving planners that maintain complex temporal models and planners that reason about the temporal resources they are using. Next, complex goals can be specified allowing the use of planners that attempt to achieve optimal results rather than blindly achieving set goals. Also, the domain will have a enough elements so that realistic conversations can be generated about the domain allowing the use of a discourse system involving plan recognition. Finally, unexpected results can occur, allowing the use of planners designed to take advantage of serendipity, and forcing the issue of mistakes and uncertain actions into the open.

3.1 Planning in the Example

For intuitions about how the issues mentioned above arise in ARMTRAK, consider the following steps in the generation of a plan for the scenario described above. Currently no planner can generate and execute such a complicated plan. ARMTRAK will, however, provide a unified domain for the entire planning community at the University of Rochester. By working together, we hope to be able to produces such plans in the near future.

1. While the system, S, is planning to move the car attached to the engine to Station B, the person, P, notifies it that she wants it to take the potatoes in a loading bin as station A to market by 4:00, and that she wants it to take the engine to the machine shop at Station A.

2. S reorganizes the priority of its goals, giving the person's goal of getting the potatoes to market the highest priority, its goal of getting the cattle car to Station B second highest priority, and the person's goal of getting the engine to the station for maintenance the lowest priority.

3. S starts generating a plan to get the potatoes to market. It recognizes that it will need to get to Station A first, so it plans a route and starts the trains moving.

4. S notifies P that it will have to move the trains at a dangerous rate of speed to make the deadline.

5. S then realizes that it knows of no way to load potatoes. It stops planning temporarily to ask P how to load potatoes.

6. P tells S how to load potatoes. This plan requires that S move a box car at Spur #3 to the loading bin and leave it there while P loads the potatoes.

7. S does not know what a box car is so it asks P.

8. P tells S that there is a box car on Spur #3.
9. S looks at Spur #3 to see what a box car is.
10. S begins planning again while beginning to move the train towards Spur #3.
11. P asks why S is pulling a cattle car, and S describes its goals.
12. P asks S to satisfy its secondary goal of getting the cattle car at Station A to Station B also.
13. S adds this new goal with a priority just above its own goal.
14. S sets the switch to move onto Spur #3, but the switch fails, and the train continues on the oval.
15. S recognizes the mistake.
16. S backs the train over the switch and sets it again.
17. S asks P to watch the switch while it passes to make sure that the switch does not fail again.

3.2 Real Time Planning

There are two aspects of real time planning: the temporal structure of the plans produced, and producing those plans in a timely fashion. The first of these aspects is a constraint on the plans produced; the second is a constraint on the planner.

Plans rely heavily on the structure temporal constraints give them. Even plans involving only a sequence of primitive actions require this sequential temporal ordering for structure. More complex interactions arise when concurrent or conditional plans are produced, or the when the planner must interact with scheduled events (as does Deviser [36]). For instance, in the above example, the system must recognize that it cannot move the cattle cars when the train is in the shop. It must therefore raise the priority of its goal over the last of the person's goals to achieve its goal at all.

ARMTRAK also has situations that can only be handled by multiple actions occurring simultaneously. Such a scenario appears in the example planning session (line 6) when the person tells the system that it must wait at the loading bin while she loads the potatoes. It is crucial that the system be able to reason about the duration of the wait at the loading bin. The system must be able to generate a plan that includes a wait of a particular time. Because it would be impossible to represent waiting actions of all different lengths, the system must be able to reason about length of time and durations of states.

In addition to the structure time imposes on plans, time imposes serious constraints on the planner. First, the planner must be able to produce an adequate plan when a plan is needed. In a dynamic domain, the changing world make plans that are produced too late useless. Second, the planner must be able to react to changing situations and abandon or alter plans that become outdated.

Plans are relevant only in that they are timely. Agents must consider the chance that planning may, by taking too much time, produce useless plans. Thus planners must be able to reason both about the time required to execute the plan and the time required to generate the plan. Brooks [10] argues that planning using world models is too costly to be performed in a timely manner. In a similar vein, Agre and Chapman [1] argue for using the world as a representation of itself. Planning can then be made less costly by considering only those things that are currently perceivable. Limiting reasoning to a few items is important because planning, Chapman [15] claims, is NP-hard. Clearly, however, humans use their world knowledge to reason about plans. These facts must be accounted for in planners that act in the real world.

In the example planning session, the person specifies a temporal constraint to the planner. It must get the potatoes to the market by 4:00. This requires that the system reason not only about the travel time between the train's current position, the loading bin, and the market, but it must also reason about the amount of time it will take to generate a plan to achieve the goal. The result of this reasoning might be the decision to start moving the trains before a complete plan is generated as our hypothetical planner, S, does in line 10.

The system cannot relinquish control of the trains when it needs to reason, so ARMTRAK requires that the system have a continuously operating control component. Timely reaction to circumstances requires intelligent control of the trains while the planner is reasoning. Because different subsystems control reasoning and acting, a trade-off occurs. The planner may exercise precise control by producing detailed plans, or it may rely on precompiled control routines. In commonly occurring situations, precise control is problematic because reasoning is costly. On the other hand, relying on precompiled routines may be catastrophic in inappropriate situations. These two types of control appear in human behavior as smooth, learned motor skills and jerky, sensory-feedback, cognition-driven activity.

In order to meet its time constraint in the example planning session, the system must start moving the trains as soon as it realizes that the train will have to be at the loading bin before it gets to the market. Thus, in line 3, the planner starts moving the trains towards the loading bin. This turns out to be a mistake, however, because it needs to pick up a box car on Spur #3 before it gets to the loading dock. It will need to rely on other routines to deal with monitoring the train while pursuing the further planning that leads it to this recognition.

When the system's plans change, or are newly defined, the activity of the control component may need to be suspended or modified. The reasoner must reason in two ways. First, it must continually revise its estimate of the value of the plan it is carrying out. Second, it must be able to deal with goals that arise while planning, not due to the elaboration of the plan, but due to perceiving a new situation. Thus the reasoner must continually be both deciding what activity it should pursue next, and the reasoning about its goals. Georgeff and Lansky[19] describe a planner that can react to new situations by halting and altering plans given new information. Firby [18], tries to build plans that react to new situations in a timely fashion by specifying a planning that consists of a many loosely-coupled simple routines. ARMTRAK provides a testbed for all of these theories of planning and activity.

The first type of revision of plans occurs in line 12. The system has organized its goals, has begun planning, and has even initiated activity towards the realization of its goals. It must now, however, reconsider its plans in the light of the person's new request. It should recognize that though the new request conflicts but little with its own plans, it conflicts in a major way with the goals of getting the engine serviced. It must therefore attempt to satisfy this new goals before getting the engine serviced. It must therefore reduce the priority of getting maintenance on the engine even further.

The second type of revision occurs when the system recognizes that it has passed the point where it should have moved onto Spur #3 (in line 15). It must be able to revise the plan in light of the situation in which it finds itself. According to the system's plan, the engine should be on Spur #3 ready to connect to the box car, but it is really still on the oval loop. It must alter its plan so that it will back up and try to move onto Spur #3 again, but it would be wasteful to throw away the entire plan. To change its plan with maximum efficiency, the system must recognize the part of the plan that fails, remove that part, and insert corrective action at that point.

3.3 Planning with Uncertainty

ARMTRAK requires agents that adapt to both anticipated and unanticipated situations. Any solution in the ARMTRAK domain must be able to adapt across the spectrum of appropriate responses. At one end of this spectrum are responses that must be made to avoid immediate disaster or to take advantage of the current situation, at the other end are responses that require much deliberation. The system takes advantage of the current situation in line 3 by starting the train. It knows enough to begin to move towards the satisfaction of its goals, and knows that simple reasoning processes are sufficient to keep the train moving in the right direction. Such a recognition, however, requires complex reasoning about possible futures. ARMTRAK situations will require combining the ability to address stereotyped problems by simple rote procedures with the ability to reason about novel problems.

The perceptual processes of the system must be well understood to be able to determine when replanning is needed. In particular, three types of information must be analyzed to determine when replanning is necessary: the cost of perception must be taken into account; the accuracy of the perception received must be considered; and the cost and probability of success given replanning in contrast to the cost and probability of success given no replanning.

This type of reasoning will be necessary for the system to realize that it needs to ask the person to monitor the progress of the train through the switch (line 17). The system will need to recognize that its own abilities have limited effectiveness in this situation in order to recognize that it needs to ask for help. It must balance the cost of asking the person to help it against the cost of failing to connect to the box car, and it must balance the probability that it can monitor the switch effectively against the probability that the person will help it.

In addition to dealing with unexpected situations, the system should be constantly monitoring the world in order to refine its knowledge. Because the system's ability to gather information is limited, it must take full advantage of what information it does gather. It should continually gather and process that information that is readily available in order to make future planning easier. Three types of incomplete information appear in the example planning session: incomplete knowledge of the current situation; incomplete knowledge of a plan to achieve its goals; and incomplete knowledge of the structure of the world.

The first type of incompleteness appears in line 17 when the system asks the person to monitor the trains for it. The system need merely rely on the person to notify it if the switch fails. This type of information can be treated as simple perception, it does not need complex reasoning to make sense of the information it gets.

The second type of incompleteness occurs when the system realizes that it does not know how to load potatoes in line 5. It can gather the necessary information by asking the person how to perform the task, but the problem has different ramifications. The system must realize that there is a technique for performing such an action, and must be able to incorporate information about how to perform the action. The system must be able, not only to incorporate new information, but to alter its model of the world to incorporate new information about how the world changes in response to its actions.

The third type of incompleteness shows up when the system asks the person what a box car is in line 7-8. Due to the difficulty of describing visual scenes, the person tells the system only where it can find a box car. The system must then process this information, infer that a box car is a type of railroad car, look on Spur #3, and combine the visual information it gathers with information it has about other types of railroad cars. If there is only one car on Spur #3, the task is somewhat simplified. The system can assume that the car on Spur #3 is a box car, and the features can be compared against the features that distinguish other types of cars. This information should be stored for future reference. If there is more than one car on the track, the problem is more complicated. The system must also determine which of the cars on the track is a box car. Probably the best solution to this problem is to ask which is the box car.

Feldman and Sproull [17, 33] discuss choosing between different possible plans by using decision theory. Such choices are possible only if there is a way in which utility can be associated with goals, and costs associated with actions required to achieve these goals. Utilities and costs cannot be associated with the goals and actions of the blocks world in a natural way. By extending intuitions about real trains to the model trains world, such an association can be achieved naturally. For a simple example, costs could be the length of the track traversed; utilities the cargo held in the cars.

3.4 Choosing among Multiple Goals

The blocks world allows the encoding of multiple goals that must be achieved simultaneously, but it is hard to see a simple way to choose among various goals if they cannot be satisfied simultaneously. ARMTRAK is different. There are certain implicit goals like "keep the train on the track" that should always be maintained. On the other hand there are those goals the system is given like getting the potatoes to market by 4:00 in line 1. In the example planning session, the goals interfere with each other. If the system runs the train fast enough to get the potatoes to market by the deadline, it is likely that the train will derail, so the system must prioritize among these goals. Should it run the train hard and risk a derailing, or should it simply fail to meet the deadline? Actually, the deadline probably is a continuum of goals; missing the deadline by a little is probably less serious than missing the deadline by a lot. How should the system prioritize keeping the train on the tracks in this continuum of goals?

The example planning session presented above also involves multi-agent planning. In this example, two agents are involved, the system and the person. The second agent further complicates the problem of prioritizing among goals. The system must be able to prioritize its own goals among those of the person communicating with it. Prioritizing goals brings up the problem of the social relationship between the system and the person. In the example above, the system occupies an employee/boss relationship with the person, but other relationships are possible, and would lead to different types of problem solving. Competition is another common social relationship between artificial agents and people. If the system and the person were competing to move their trains around a partially shared track, the negotiation between them would have a different character.

In the example scenario, the two agents are cooperating to achieve mutual goals. In line 5, when the system asks the person how to get the potatoes to market, multi-agent planning is occurring. Part of the system's plan requires gathering information from the other agent, so the systems plan is contingent on activity by the person. If the person does not answer the goal cannot be achieved; asking the person is a necessary part of achieving the goal of taking a load of potatoes to the market.

The problem of prioritizing goals involves two different types of reasoning. The system must be able to choose between goals that it will achieve, and it must be able to choose among the means of achieving the goals. Deciding which goals to achieve will involve determining which goals are contradictory. Deciding the means to achieving the goals will involve calculating the relative merits and costs of achieving the goals. The merit of the goal, should include the chance that the goal may fail due to unexpected circumstances.

The problem of choosing among goals arises immediately in the example. The first thing the system does after receiving instructions from the person in line 1 is organize the priority of its goals. (Line 2) Its own goal of moving the cattle car to Station B will be impossible if it achieves all the person's goals before it achieves any of its own, so it must not simply accomplish the person's goals before attempting to achieve its own.

In order to choose goals appropriately, the system will need a model of the person's goals. The system should be able to recognize that the person's goals probably involve keeping the train on the track. The system could reason that if the train jumped the track, the person's goals cannot be achieved, but several reasoning steps can be saved by recognizing through simple calculations that excessive speed will be necessary. As it does in line 4, the system can then notify the person of the necessity for excessive speed before the situation becomes even more critical.

The problem of choosing among the means to achieving a goal arises in line 17. The system is unable to monitor train movement through the switch, so it asks the person to do it. The system could simply continue to run the train through the switch until the switch worked correctly. It could determine that the switch has indeed worked when the train did not reappear on the oval where it could see it. This solution is much more costly than asking the person to watch the switch, however, and the system should be able to reason that, since the goals it is pursuing are those of the person, the person should be invested in achieving them in a timely fashion. Thus, there is benefit in asking for help.

3.5 Current Research

At present, research into all the issues mentioned above is ongoing at Rochester. Much of this work is structures around Allen's two level architecture for planning [3, 5]. This architecture involves a reactive control system and a complex plan reasoner whose communication is limited. A plan reasoning system is being developed using the timelogic system [27] which allows one to write programs that reason about such constraints. Timelogic can be accessed through Rhetorical [7], a first order logic knowledge representation system. One of the crucial aspects of this project is dealing with action abstraction. Tennenberg [34] has developed a formal model of abstraction in STRIPS-type domains and is currently extending this work to non-linear and temporal planning frameworks.

Weber [37] has built a system that uses heuristics to find the most appropriate information using a parallel algorithm for calculating expectations. This system provides a first cut at the problem of relevance. Hartman [22] is studying the problem taking into account the cost of planning while planning.

Reasoning under uncertainty is of primary concern at Rochester. Kyburg's epistemological probabilities [24, 25] provide one of the primary candidates for representing uncertain information. Tennenberg and Weber have used this system to provide a solution to the frame problem [35]. Loui [28] has built a reasoner that uses epistemological probabilities and has developed decisions theory for this system. In addition, Feist is studying the problem of planning to monitor plans that might fail due to uncertainty.

The problem of choosing which among a group of goals to solve is less well studied. Pelavin and Allen [31] provide groundwork for the solution of this problem by suggesting a formalism that allows one to reason about actions that are the result of activity other than the reasoning agent's. Kautz's [26] work in plan recognition provides the foundation for giving a system the ability to infer plans from language. This ability is important for multi-agent planning.

4 Natural Language

SHRDLU [38] used the blocks world as a domain for natural language, and the capabilities this system were suited to the constraints imposed by the domain. The domains in which subsequent natural language systems are tested have, for the most part, made similar simplifying assumptions. These assumptions are similar to those made in planning domains. They are:

1. only the system has effects on the domain of discourse,
2. the only means of communication is through the console,
3. the system and the person have the same goals, and
4. the system has access to complete information about the domain.

ARMTRAK is a useful domain for examining some aspects of communication. For example, plan recognition [2, 26] is frequently used in natural language understanding systems to resolve semantic ambiguities. The need for plan recognition arises only under circumstances that are complex enough to require disambiguation.

When it is implemented on the Rochester Robot, however, ARMTRAK becomes a much more interesting domain. The crucial difference between such an implementation and other domains for experimentation in natural language systems is interaction with the real world. Such interaction allows experimentation in natural language in three areas.

1. Activity takes place during the conversation,
2. the system may have only partial knowledge of the world, and
3. both the system and the human interlocutor use language to achieve their respective goals.

Interaction with the real world makes ARMTRAK a particularly rich domain for investigating man-machine communication using natural language. On the other hand, the ability to reduce the domain to few predicates and constants makes incremental experimentation possible. The minimal system, described above, consisting of one, two or three cars, a single engine, and a circular track with three spurs is rich enough to engender the example conversation presented in the next section. Though the conversation is complex, the number of items actually referred to is small, reducing the problem of real time inference. The problem of real time inference can also be addressed in ARMTRAK by increasing the number of cars in the system. Problems introduced by complex world knowledge can be investigated by associating different characteristics with the cars. For example, in the following conversation, the system asks what kind of cars are good for carrying potatoes. To pose such a question, the system must have concepts of railroad cars and potatoes. Ever more complexity can be introduced into the model by adding ever more complex information about trains and their behavior.

4.1 A Dialogue about the Example

The following constructed dialogue gives a flavor of the rich conversations possible in this domain and is provided to aid the reader's intuition. Specific problems that arise in this proposal will be discussed following the example.

The situation in which the dialogue occurs is the same as the situation in which the planning session occurs. That is, the person, P, and the ARMTRAK system, S both have initial goals that are not mutually known. The person's goal is to get a load of potatoes from the loading bin at station X to the market near station B and, once this is done, to have the engine maintained. The system's goal is to get the cattle car 37 that is attached to the engine to station B. *This situation is summarized in figures* 4 *and* 5.

1. **P**: I'd like you to take a load of potatoes to the market by 4:00.
2. They are in the loading bin at station A
3. When you're done, you should return to the engine shop for maintenance.
4. <Trains start to move>
5. **S**: I'm going to have to move the trains too fast. They might derail.
6. **P**: Don't worry.
7. **S**: I've never loaded potatoes before.
8. How do I do that?
9. **P**: I'll operate the loader at station A.
10. You back the car into the loading bin,

11. open the car doors, and

12. leave it for about 20 minutes.

13. **S**: What type of car is best for potatoes?

14. **P**: Any car but a tanker or a cattle car will work, but a box car is best.

15. **S**: What's a box car?

16. **P**: There's one on spur #3.

17. **S**: OK.

18. I'll pick up the box car on spur #3.

19. **P**: By the way,

20. You're pulling a cattle car already.

21. **S**: I know

22. I need to drop it off at station B.

23. I'm going to do that after I drop the potatoes off.

24. OK?

25. **P**: Oh.

26. In that case,

27. while you're at station A, can you pick up another cattle car that also needs to go to station B?

28. <The switch to Spur #3 is set incorrectly and the train moves past the turn off>

29. **S**: Oops, I missed the turn off.

30. <The trains moves back across the switch to try again.>

31. **S**: Is the switch set correctly?

32. **P**: Yes, the switch is OK.

33. You're about to cross it now.

34. <PAUSE>

35. You're past it.

36. **S**: Thanks. I'm stopping the engine now.

37. I can get your other cattle car with no problem.

38. I'm backing up.

39. Can you let me know when I hit the box car?

40. **P**: OK.

41. OK. You've got it.

42. There's nothing else attached to it, so you're ready to go.

Natural Language is used for many distinct purposes in the dialogue. In particular, it is used as a form of generalized perception, it is used for cooperative problem solving, and it is used to monitoring the plan's execution. In addition, it signals the status of the conversation explicitly as the dialogue progresses to insure effective and efficient communication. Finally, it is used to resolve and express uncertainty. The following sections discuss each in turn.

4.2 Talking about Plans and Goals

Language may be used as a generalized form of perception in the sense that it is used to acquire knowledge about the external world much in the way the vision system is used. Both involve observations and inference. The vision system provides certain primitive functions, and the system uses this information to infer the state of the world. Similarly, the language system provides an analysis of the structure of the incoming sentence, and the system uses this information to infer the state of the world again (via the recognized intentions of the speaker). Consider two examples. Say the system has a goal to find a red train-car. It might plan to scan the track with the cameras until something red is seen, and then attempt to identify this as a car. Alternatively, it might ask the person if she knows where a red car is, and use this information to form a plan directly, or to reduce the search needed by the camera. As another example, consider the system as it attempts to move an engine up to a car. Again, it might use the vision system, or it might ask the person to tell it when it is touching (as in lines 39-41 above). If the car is in an area that is out of the system's range of sight, then only the latter might be a viable plan.

Language is also used as part of the problem solving process, either so that the two agents may agree on a joint plan, or so that one may assist the other in formulating a plan. For example, in lines 9 -12 above, the person suggests a joint plan, where it operates the loader, and the system moves the car in and waits. As another case, if the system has a goal of getting a load of grain to a certain location, the person might advise it to avoid a route involving a bad track. When the two agents have individual goals of their own, deciding on a plan becomes a negotiation as well. For example, lines 21- 29 involve a reformulation of the plan so that the goals of both the man and machine can be accomplished within the same plan. Language may also be used to monitor the execution of the plan, and to suggest quick solutions to expected events that occur. An example of this was given above when the person notified the system when the engine was near a certain car. Other examples are reports of unexpected circumstances (the switch ahead is set wrong), suggested solutions (trying slowing down and activating the switch again), and as an aid to real-time plan recognition. For example, sentence 19 is generated in response to recognizing an act that is not part of the plan that the person expected. This leads to the clarification of each agent's goals and a revised plan.

The system will need a model of both its own knowledge and its human interlocutor's knowledge. In sentences 8-9 the robot asks how to load potatoes. Because it cannot do it for itself, it chooses to use language to achieve its goals. It also needs a model of the human's abilities to recognize that the human will be able to monitor the switch for it in sentence 31.

4.3 Modeling the Communication Process

Because activity is ongoing during the dialogue, the amount of time taken to convey information is important. To increase the efficiency of communication the system will need to perform tasks like parsing, lexical lookup, and reasoning in parallel. In addition, the system will need to be able to reason about the plans and goals of the person to optimize its use of words. Reasoning about the person's intentions allows it to use a phrase as a marker as it does in line 17.

Another type of efficiency occurs in sentence 16, when the robot successfully resolves a "one" anaphora referring to an object mentioned in the previous sentence. Because there is two way interaction between the robot and the human, such references will be common.

Interactions at the discourse level also insure that the person and the system communicate efficiently. In particular, there are specific utterances that function solely to maintain the discourse interaction. For example, there are three instances of the utterance "OK" in the dialogue, which show different uses as described by Grosz [20]: Utterance 17 signals that S understood and believed P's answer to a question. Similarly, utterance 40 signals that P has agreed to S's request, and 41 signals that a certain stage has been reached in the plan being executed. In other cases, acknowledgment occurs implicitly: For example, utterance 24, "OK?" is a request for an acknowledgment. Even though P's response 25-27 doesn't explicitly acknowledge as requested, it implicitly acknowledges the situation since P did not object to it. The planning model of the discourse interaction to provide a precise account of this type of behavior as well as the problem solving behavior mentioned above. Grosz and Sidner [21] examine aspects of discourse that appear not to be a direct result of planning behavior, and such a model should eventually be incorporated with the results of this work.

In addition to discourse markers, there are several indirect speech acts in the dialogue. In particular, we see requests that are made by utterances such as:

- "I'd like you to ..." (1)
- "You should return ..." (3)
- "Can you let me know when ..." (39)

Each of these sentences could be interpreted in a literal manner in some settings, but appear to be effortlessly identified in their indirect readings in this dialogue. Plan-based models have been used to handle this problem for over a decade now [6], although these models were not sensitive to different subtleties of phrasing. Hinkelman and Allen [23] proposed a model that allows constraints from both the form of the utterance, and constraints from plan-based reasoning about the context, to be combined to form an efficient and general model of speech act interpretation.

4.4 Conversation given Partial Knowledge

Because the world about which the person and the system are talking is larger than either is able to deal with completely, the system must be able to model its own and the person's abilities. The ability to model the person's ability will give the system the ability to determine when it is able to glean information about the world from the person. The ability to gather information from the human requires a flexible parsing strategy and lexicon.

Parts of the dialogue involve several levels of clarification subdialogues. Utterances 7-8, for instance, show that S has understood the request but doesn't have the knowledge required to perform the act (namely, loading potatoes). After P gives instructions 9-12 to clarify the initial request, S asks for yet further clarification 13. Utterance 14 provides this additional clarification and 17 signals that the initial request 1 has been understood and the goal accepted. Of course, 17 could have been said only to acknowledge that P now knows what cars are best to carry potatoes, or it could be to acknowledge that P now knows how to do the task, but has not yet accepted it. It is only from the continuation, 18, that the appropriate interpretation can be identified. Again, the planning model should provide a detailed account of why these interpretations are all possible interpretations, and how the correct one is identified.

Because conversations in the ARMTRAK domain are not limited by a world model to which the natural language subsystem has access, the system needs to be able to extend its lexicon. At sentence 15 the robot notifies the human that "box car" is a word that is not in the lexicon. The human gives the robot information about the cattle car in sentence 16. The robot must add this information to the lexicon. To allow inserting new words into the lexicon, the parser must guess the features of words not in the lexicon. Much information is encoded in the position of words in the sentence in English, and this information should be available to the lexicon when it is inserting words. To do this, the parser will have to be able to return partial information. The features of this word should be garnered from the context. The robot knows that the new phrase is a noun because it has a definite article attached to it. Moreover, it should realize that the new phrase refers to some sort of railroad car by analogy with tank car and box car. This ability will require complex interaction between the parser, the lexicon, and the reasoning system.

Using ARMTRAK as a domain for investigating issues in natural language processing allows serious investigation into issues of semantic interpretation. Because the robot has access to the real world, issues of the meaning of words can be investigated experimentally. Issues such as the relative merits of Montegue logic and situation semantics [16] can be investigated in a domain where words actually refer to objects that exist outside the realm of discourse.

Semantic interpretation will need to be able to connect words with sensations. For example, when the human says "box cars" to the system, the system will need to be able to distinguish those objects in novel situations. The connection between words and sensations is made even more problematic by the uncertainty inherent in sensation. All information gathered from the real world is contingent and uncertain. Natural language, on the other hand, seems to be less so. Some means of deriving beliefs that can be reported in natural language must be developed.

The system will also need to be able to report probabilities. In sentence 5 the system reports probability by saying "There's a good the chance train will derail." The system will have the ability to compute the exact level of its belief that the train will derail, but it must be able to translate this number into a phrase the human will understand. Such a translation is not straightforward. There are probably no simple thresholds determining the probability values that correspond to words used to hedge statements.

The system also needs a model of the capabilities of the trains so that it can report that there is a danger of the trains derailing in sentence 5. Moreover, the decision to make this report requires the system recognize that the sequence of actions necessary to rerail the trains is complicated and that the human may wish to change the current goals to avoid this disaster.

4.5 Current Research

Much of the work in natural language grows out of the Discourse System Project [4]. The software developed by this project consists of six modules communicating through a blackboard. These modules are: a lexical module, a syntactic module, a semantic module, a reference module, a speech acts module, and a tense and aspect module. The lexical module accepts and identifies words. Its analysis returns the input word, the suffix information, and the root dictionary entry. The syntactic module is not yet implemented, but its task will be to parse input sentences. Poesio is currently working on a parser that may provide the necessary functionality. The semantic module transforms the syntactic information produced by the syntactic module into a logical form. This transformation generates information that is implicit in the words used from information gathered from the dictionary and information gathered from the word position. The reference module associates world knowledge with the logical forms produced. To do this it interprets the conceptual content of expressions and associates them with the objects to which they refer. The speech act module uses Allen and Hinkelman's [23] technique for interpreting speech acts. The temporal module analyzes the temporal information inherent in the discourse.

The Discourse System Project used the tasks faced by a reference librarian as its domain. It quickly ran afoul of the vast amount of knowledge necessary to deal with requests for information from the library. ARMTRAK is a domain in which conversations as complex as those of the project could take place. This domain does not, however, require an inordinate amount of world knowledge. Almost all the software developed for the discourse project can be used in the ARMTRAK domain.

5 Active Vision

In the past, much work in vision and robotics has concentrated on isolated problems, which, it was hoped would provide general solutions which could then be combined in a flexible system. Constraints imposed by real problems made much of this work difficult to apply in practice. Like the blocks world in planning, these isolated problems have generated many useful results, but, because of their isolation, they have also lead to neglect of some important aspects of sensing and acting. Some of the simplifications that arise from isolated study of vision and robotics are:

1. the task being studied need not coordinate its activity with other tasks,
2. the task can be arranged so that extraneous problems disappear,
3. temporal constrains can be ignored,
4. the task is never impossible, and
5. everything necessary for the task is always available.

Building ARMTRAK will provide a testbed for work in vision and robotics. Because ARMTRAK must satisfy a wide range of tasks, the usual simplifying assumptions cannot be made. As a testbed for active vision, ARMTRAK will allows experimentation in:

1. sensing and control under temporal constraints,
2. sensing and control under sub-optimal circumstances, and
3. choosing sensing and control tasks appropriate to overarching goals.

Because the trains move regardless of the robot, the designers control and sensing routines are faced with real time constraints. For example, it is not sufficient to merely perceive an impending collision, the collision must be recognized before it occurs.

In the real world, agents are faced with situations in which their perceptions or actions are somewhat degraded. A person, for example, can recognize and manipulate objects even when the objects are partially obscured and the person is wearing gloves. ARMTRAK provides the ability to perform experiments in similar situations. Parts of the track may be obscured, or inaccessible to the robots control. Many experiments in robotics involve mobile robots. These robots have the advantage of moving through the real world, and therefore actually encounter the problems faced by people. ARMTRAK takes a different attack. Instead of allowing the robot to range freely in the real world, the real world is shrunk so that the mobility the robot does have is sufficient. This allows experiments to proceed incrementally. Initially strong simplifying assumptions can be made allowing initial testing of primitive routines. As more realism is introduced into the layout, these initial assumptions can be relaxed.

People have many methods of effecting and sensing the world. Given a particular goal, a person chooses among the means that cause the desired effect for the one most appropriate to the circumstances. The choice of an appropriate technique for achieving a goals involves weighing the cost of applying that technique, the cost of failure, the probability that the technique will succeed, and the benefit that will result from success. Much work in vision and robotics does not address this problem because either only one technique is being studied, or the goals are fixed before hand. Because the robot's goals will be driven by the planner, fixing the goals at the start is not feasible. Similarly, because the robot will face a wide variety of tasks, it is unlikely that a single technique will suffice for all.

5.1 Examples of Sensing and Control

Rochester's binocular robot head is one of a new generation of active vision systems being developed around the world. It mounted on a six degree of freedom arm. The two cameras are on a common tilt platform, and have independent pan axes. The arm can position the camera anywhere over the floor within a working radius of approximately 8 feet. The hardware is capable of motions comparable to primate performance (about 1 m/sec head translational velocity with less than 1 mm. positioning accuracy, 200 deg/sec head rotation, and 300 degrees/sec camera rotations with .14 degree positioning accuracy). The camera controllers are capable of supporting full-speed gaze-shifts to random directions at a rate of 5/s. The aperture, focal length, and focus of the cameras are not yet computer controllable. The video output is processed by a Datacube MaxVideo pipeline-parallel image processing system that can do many low and intermediate-level vision operations at 30Hz (video frame rate). The host computer for the system is currently a Sun/3 computer but will soon be a 24-node Butterfly Parallel Processor (BPP). Each node of the BPP is an M68020 processor with M68881 floating point coprocessor and 4MB of memory. A fast communications switch allows the processors to share memory or pass messages efficiently. An additional a real-time operating system will control the dextrous hand and perhaps other peripherals over the VME bus. LISP planning programs communicate with robot applications code over the ethernet. Fig. 6 shows the hardware

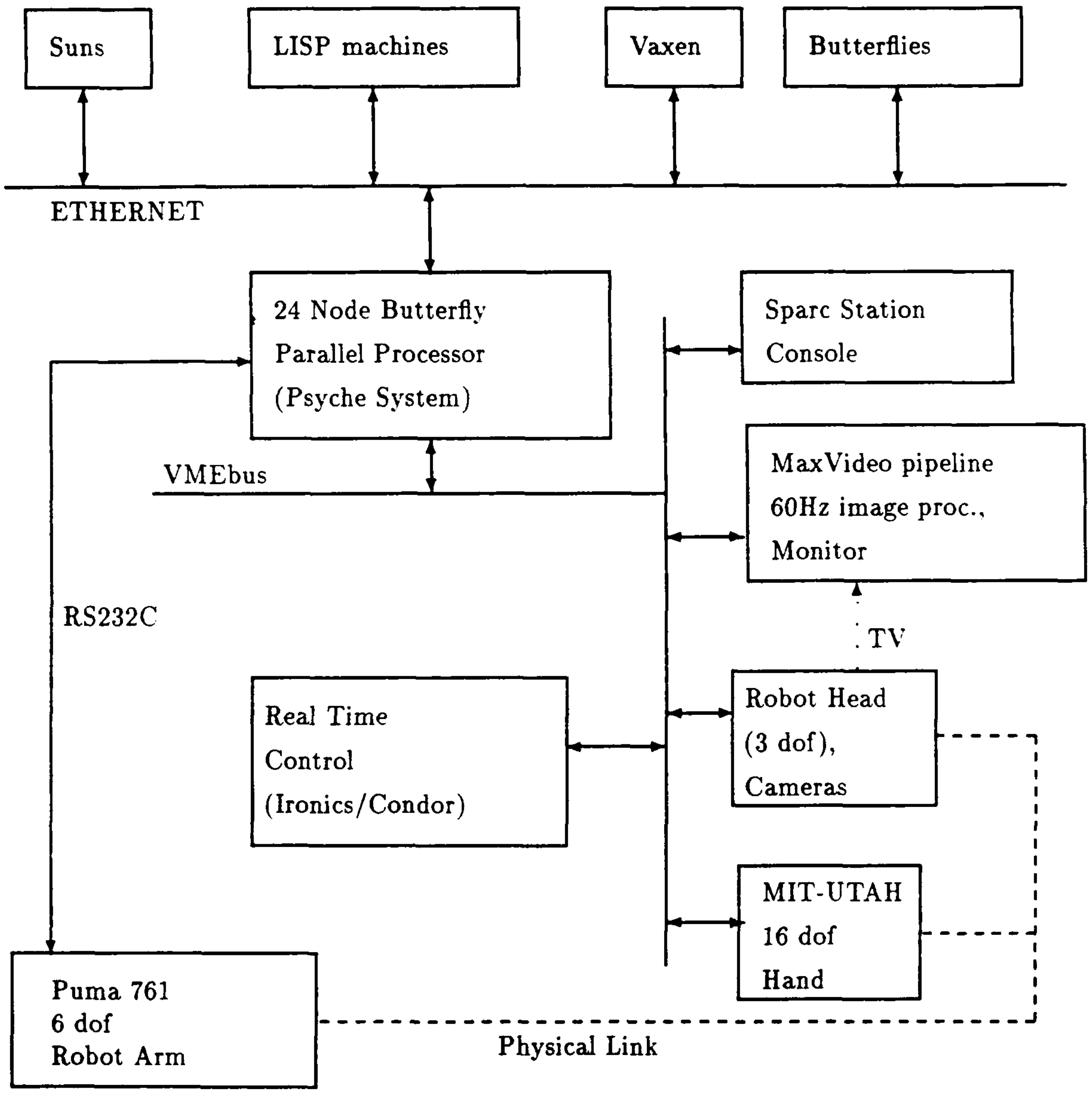
Suns
LISP machines
Vaxen
Butterflies
ETHERNET
24 Node Butterfly Parallel Processor (Psyche System)
Sparc Station Console
VMEbus
MaxVideo pipeline 60Hz image proc., Monitor
RS232C
TV
Real Time Control (Ironics/Condor)
Robot Head (3 dof), Cameras
MIT-UTAH 16 dof Hand
Puma 761 6 dof Robot Arm
Physical Link

Figure 6: Architecture of the hand - eye system as of mid-1990. As of late 1989 the central host is a Sun (though the Butterfly is running some applications), and the hand is still in the acquisition phase.

organization we plan to attain by mid-1990.

From this hardware, a sensorium will be built so the robot head is positioned by the Puma robot. Visual information from the cameras will be sent to the MaxVideo for initial processing, then to the Butterfly for more sophisticated processing. The results of the more sophisticated processing will, in turn, be used to guide the positioning of the Puma.

As an example of the type of processing required of the system consider the problem the robot will face when it discovers that it does not know what a box car is, and that there is and example of a box car on Spur #3.

1. The system glances over at Spur #3 to see if it can distinguish a type of railroad car that it does not recognize
2. It cannot find such a car, so it moves the head over so it can get a better view of Spur #3.
3. As the head moves, it monitors its view, and stops moving as soon as it sees a car whose type it doesn't recognize.
4. The system determines how the box car should be indexed among known railroad cars types.

5.2 Temporally Constrained Sensing and Control

For the tasks faced by the ARMTRAK robot, serial hardware is too slow. Two types of parallel hardware are available to speed the processing of visual processing. Simple, frequently repeated algorithms can be implemented on the Datacube processor. Because the Datacube is a special purpose machine, however, only a few techniques can be implemented this way. The Datacube cannot be reconfigured while a task is running. The Butterfly multiprocessor, on the other hand, is a general purpose multiprocessor. It can be reconfigured as the task demands. Because it is a general purpose machine, however, it is not as fast as the Datacube for vision processing.

To be able to deal with the example problem, the system must be able to take care of both types of parallelism. The visual primitive from which the system constructs its visual representation of the box car, should be implemented as MaxVideo routines. The unprocessed image contains too little information that is constant over different views. In addition, the processed information arriving from the MaxVideo should be combined using the Butterfly. The Butterfly allows several processors to coordinate their activities on a single image.

Time itself must enter into the description of processes and the value of computations to the system. The sensor control system must be opportunistic to take advantage of unexpected information. In addition, it must be realistic to abandon the search for information as time constraints take over. Thus the management of sensors parallels the management of resources elsewhere in the planning system of which it is a part, and itself must be mirrored in the operating system that provides the foundation for the integrated system. Just such a system is being built at Rochester: the Psyche operating system [32] provides the necessary functional and programming environment support. one active vision program, Juggler, has already been ported to Psyche.

5.3 Sensing and Control under Sub-optimal Circumstances

The temporal constraints require that the system deal with information that is sub-optimal in some aspects. If it chooses to glance at Spur #3, the box car will be sufficiently small that some detail will be lost. Those routines which rely on this detail will not give good information. The system must be able to make the most of the information it is able to gather in the time it has.

Moreover, the system's performance should degrade gracefully in the presence of occlusions and clutter. If there is only one car on Spur #3, the system should be relatively certain that it knows what a box car is. If, on the other hand, there are many railroad cars on Spur #3, it should still be able to guess that it is one of those that it cannot identify.

The robot's control routines should also be able to deal with sub-optimal situation. It may be impractical in some circumstances to keep the robot calibrated. The system should be continue to function in these circumstances.

5.4 Choosing Appropriate Tasks

The problem of choosing appropriate tasks occurs twice: during design and at runtime. During design a flexible, computationally tractable set of primitives must be chosen from which more complex behaviors can be constructed. At runtime, the robot's control mechanisms must construct the behaviors from the primitives provided.

ARMTRAK motivates several visual capabilities closely related to the ones we have studied. For instance, recognizing a moving railway car will involve tracking it (stabilizing its image). Another useful gaze-control primitive is redirecting the gaze toward a known point, possibly adjusting the head position to get a clear view. Dynamic models of the train and robot will allow trajectories to be calculated that achieve synchronous relationships between moving trains and the head. The timely achievement of visual tasks raises issues that parallel those of "real-time" planning. A repetoir of such behaviors must be developed that may be used to complete whatever activity a plan calls for.

In addition to this repetoir of activities, the system must be able to compose more complex behaviors from them. It must be able to decide the most useful place to look next given its task and its abilities. In the example, the robot must decide where Spur #3 is and how be to gather the necessary information to distinguish box cars.

5.5 Current Research

Part of the ongoing research effort in the robotics and vision laboratory is to endow the Rochester Robot with an increasingly sophisticated visual repertoire, especially capabilities needed to support tasks in visually dynamic environments.

Recent work with the Rochester Robot produced several implementations of potential basic components of a real-time gaze-control system [8, 11, 9, 30]. These components included basic capabilities of target tracking, rapid gaze shifts, gaze stabilization against head motion, verging the cameras, binocular stereo, optic flow and kinetic depth calculations. For instance, tracking is accomplished with a special-purpose board that correlates the image with an 8×8 template of the pattern to be tracked, and a board that converts over-threshold correlation peaks to image (x, y) coordinates, that are then converted to camera pan and tilt commands. To avoid false matches with the unnormalized correlation operation, preprocessing is done to remove low (or zero) frequency brightness variations, and the best match may not be too far from its position in the previous frame. Vergence signals arise from a cepstral filter (similar to phase correlation) implemented with the integer fast Fourier transform on a digital signal processing chip, producing a global disparity measure that is then converted into a camera pan command. The computation takes approximately 40ms.

Current work is addressing the cooperative interaction of visual and motor behaviors in the robot hand-eye setup. For example the "Juggler" project is attempting to keep a balloon in the air with a racquet, using visual input. The cooperation of basic "reflexes" in a gaze control system has been studied only in simulation [13, 12], though some implementation work is proceeding.

Ultimately the robot may be considered a form of agent itself, with its own hierarchy of controls and repertoire of skills. Our plan is to design the intelligent robot agent out of component skills that interact loosely [29]. Other skills can be built by observing the effects of combined primitive actions and improving the collaboration, synchronization, and mutual awareness of the primitive actions to develop a more efficient implementation of the desired behavior.

6 Currently Implemented

Two versions of ARMTRAK have been implemented: a simulation and a set of trains coupled to the sensorium associated with the Rochester Robot. The simulation allows rapid prototyping of planners and experimentation with problems posed by different layouts. Because building model train layouts requires an investment of effort, the availability of the simulation allows users of the systems to perform quick experiments to answer specific questions. On the other hand, simulations invariably involve simplifying assumptions. The availability of the real trains and the sensorium provided by the vision lab insures that the results obtained using the simulation are accurate. The simulation will allow easy experimentation; the real trains will insure that these experiments are honest.

6.1 The Simulation

The ARMTRAK simulation displays the following characteristics:

- The information describing the simulation is maintained as a relational database allowing easy modification of the possible queries.

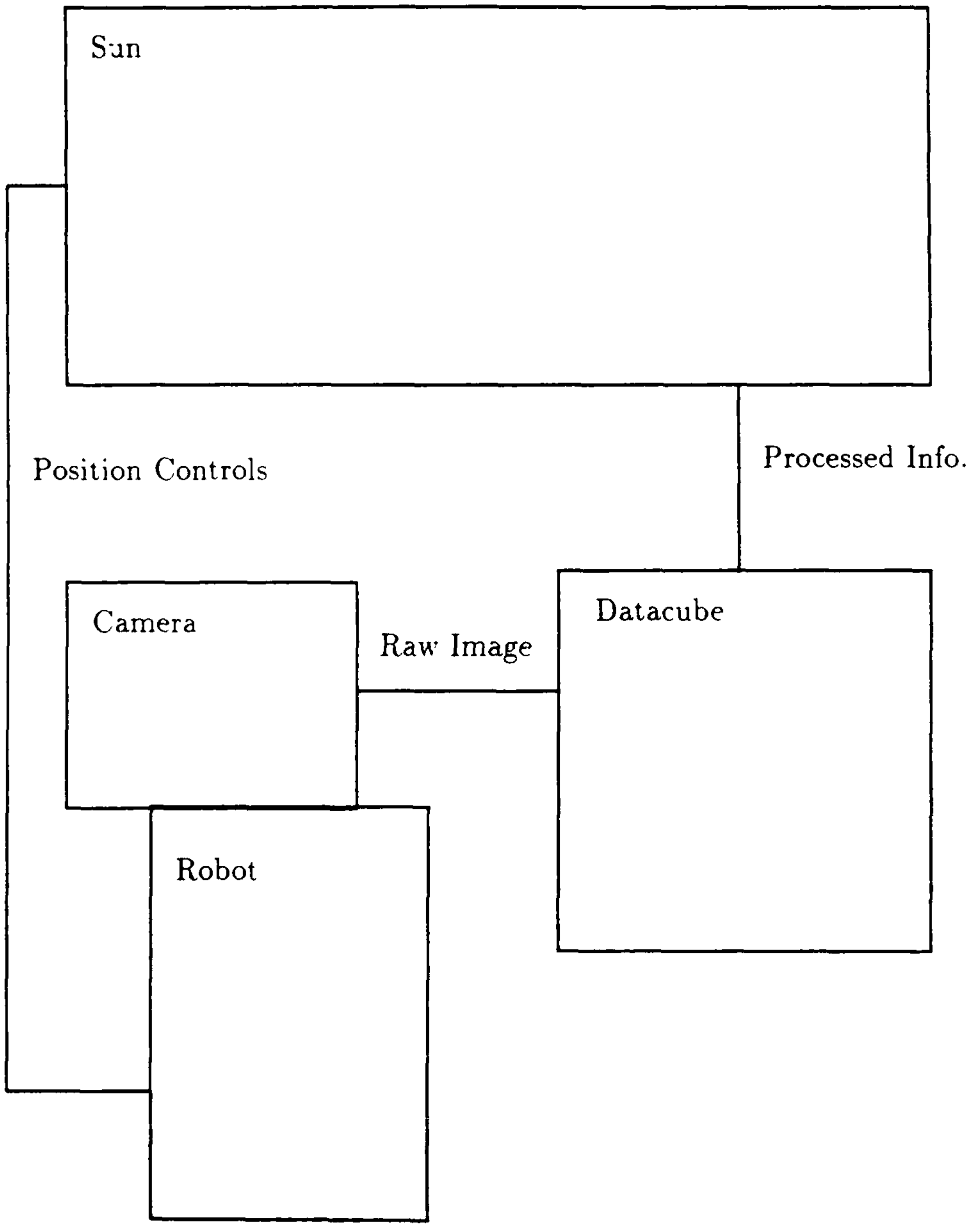

Figure 7: Overview of ARMTRAK Sensorium Architecture

- Planners link to the simulation through a series of library functions that return values on query, or change the state of the simulation on command.

- The communication between the simulation and the planners is maintained by stream based sockets allowing the use of planners on other machines. Separating planners from the simulator also insures that planners built using it will work with the real model trains.

The overriding concerns in the development of the simulation is that it accurately models the real model trains, and that planners developed using the simulation can easily be applied to the real trains.

In the ARMTRAK simulation, both the trains and the sensorium are simulated. The trains are simulated by a database that is updated periodically by a set of real-time control routines. The sensorium is simulated by a set of database queries. A library of access functions is provided through which the user accesses the simulation.

The ARMTRAK simulation consists of five modules and a library of functions. The modules are the input module, the database module, the real-time control module, the communications module and the error and debugging module. The input module takes an initial description of the layout and produces a memory resident database that can be manipulated by the real-time control routines. The database modules defines the database schema and provides routines for manipulating it. The most important of these routines are select, project and join. Using these three operations any element in the database can be accessed. The real-time control module updates the database periodically providing the illusion of movement for the simulation. Communications between the user and the simulation are performed by routines in the communications module. The communications module reads messages sent out by the library routines and passes them to the appropriate module for execution. The error and debugging module contains routines that catch errors and notify the users of problems. Also included in this module are routines useful in debugging the program as it was developed. All these modules are tied together through the ARMTRAK module.

The library consists of a set of routines that allow the user to alter and test the state of the simulation. These routines are linked into the user program and provide a means of setting up communications. They also provide a restricted set of command that can be sent to the simulation.

The ARMTRAK simulation runs on both Sun/3s and Sun/4s. It can be accessed through library routines on Sun/3s, Sun/4s and Symbolicses. Library routines for the Sun/3s and Sun/4s have been implemented in both C and Lisp. The library routines on the Symbolics are implemented in Lisp. Though the connection to Lisp, the ARMTRAK simulation is already being used as a testbed for planners.

6.2 The Rochester Robot Version

The version of ARMTRAK that uses the Rochester Robot is much less complete than the simulation. It has primitive versions of vision routines to satisfy the three ARMTRAK queries, but the commands are simulated by hand. The vision routines are able to recognize the existence of a moving train in its field of view and are able to determine the state of a switch in its field of view. It also knows the positions of the switches so it can position itself to find them. The train controller has been wired so that the setup (the trains, switches, and decouplers) can be operated from outside the robot room.

This implementation requires four pieces of specialized hardware: the layout, the head, the arm, the MaxVideo box, and Ophiuchus. The layout is a set of HO gauge model trains connected by a cable to a controller located outside the robot room. The head consists of two cameras mounted on a platform that can move either camera's yaw independently or both camera's pitch simultaneously. The arm is a Puma 760 connected to a Val controller implemented on top of an LSI-11. The MaxVideo box consists of a set of image processing cards connected to a host computer by a VME bus. The cards necessary for ARMTRAK are a Digimax, a RoiStore 2048, a MaxMux, and two FeatureMaxes. Finally, Ophiuchus is a Sun 3/280 connected via VME bus to the MaxVideo cards and the controllers for the eye motors, and via serial line to the Val arm controller. Figure 7 shows the hardware setup for this version of ARMTRAK.

This version of ARMTRAK comprise six modules: the error module, err; the main module, ar; the communications module, com; the MaxVideo module, mv; the head control module, hd; robot control module, pu; and the connected component module, blobs. The error module writes ARMTRAK error messages. The main module uses the other modules to implement the ARMTRAK functionality. The communications module provides communication over the ethernet to Ophiuchus so the predicates can be used on other machines. The MaxVideo module sets up and controls the MaxVideo cards. The head control module sets up and controls the eye motors and maintains a data structure describing the current configuration of the eyes. The robot control module initialized the Val controller and controls the robot through the serial line interface. The connected component module analyzes feature lists generated by FeatureMax for blobs.

7 Acknowledgements

This work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, under contract no. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC). Additional support was provided by ONR/DARPA research contract no. N00014-82-K-0193, ONR research contract no. *N00014-80-C-0197, and U.S. Air Force research grant no.* AFOSR-89-0222.

References

[1] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. *AAAI*, 5:268–272, 1987.

[2] James Allen. *Natural Language Understanding.* Benjamin/Cummings, New York, 1987.

[3] James Allen. Formal models of reasoning about plans. In preparation, 1989.

[4] James Allen, Stephane Guez, Louis Hoebel, Elizabeth Hinkleman, Keri Jackson, and David Traum. The discourse system project. Computer Science 317, University of Rochester, 1989.

[5] James F. Allen. From reasoning to acting. In preparation.

[6] James F. Allen. *A Plan-Based Approach to Speech Act Recognition.* PhD thesis, University of Toronto, Toronto, Canada, 1979.

[7] James F. Allen and Bradford W. Miller. The rhetorical knowledge representation system: A users manual. Technical Report 238, University of Rochester, 1989.

[8] D. Ballard. Animate vision. In *International Joint Conference on Artificial Intelligence 1989*, pages 1635–41, August 1989.

[9] D. H. Ballard and A. Ozcandarli. Real-time kinetic depth. In *Second Int. Conf. on Computer Vision*, November 1988.

[10] Rodney A. Brooks. Intelligence without representation. *Workshop on Foundations of Artificial Intelligence*, pages 1–21, 1987.

[11] C. M. Brown. The rochester robot. Technical Report 257, University of Rochester, September 1988.

[12] C. M. Brown. Gaze controls cooperating through prediction. *Image and Vision Computing, in press*, ??(?), ?? 1990.

[13] C. M. Brown. Gaze controls with interactions and delays. *IEEE Transactions on Systems, Man, and Cybernetics*, IEEE-TSMC20(2), March 1990.

[14] Christopher Brown, Dana H. Ballard, Timothy G. Becker, Roger F. Gans, Nathaniel G. Martin Thomas J. Olson, Robert D. Potter, Raymond D. Rimey, David G. Tilley, and Steven D. Whitehead. The Rochester Robot. Computer Science 257, University of Rochester, 1988.

[15] David Chapman. Planning for conjunctive goals. *AI*, 32:333–377, 1987.

[16] Robin Cooper. Meaning reprresentation in Montague grammar and situation semantics. *Computational Intelligence*, 3:35–44, 1987.

[17] Jerome Feldman and Robert Sproull. Decision theory and artificial intelligence II: The hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[18] R. James Firby. An investigation into reactive planning in complex domains. *AAAI*, 5:202–206, 1987.

[19] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. *AAAI*, 5:677–682, 1987.

[20] B. J. Grosz. The representation and use of focus in a system for understanding dialogues. In *IJCAI*, pages 67–76, 1977.

[21] B. J. Grosz and C. Sidner. Attention, intention, and the structure of discourse. *Computational Intelligence*, 12:3, 1986.

[22] Leo Hartman and Josh Tenenberg. Performance in practical problem solving. In *IJCAI-87*, pages 535–540, 1987.

[23] E. Hinkelman and J. Allen. Two constraints on speech act ambiguity. In *Proc. ACL*, pages 212–219, 1989.

[24] Henry E. Kyburg Jr. *The Logical Foundations of Statistical Inference.* Reidel, 1974.

[25] Henry E. Kyburg Jr. The reference class. *Philosophy of Science*, 50:374–397, 1983.

[26] Henry Kautz. *A Formal Theory of Plan Recognition.* PhD thesis, University of Rochester, Rochester, NY 14627, 1987.

[27] Johannes A.G.M. Koomen. Reasoning about recurrence. Technical Report 307, University of Rochester, 1989.

[28] Ronald P. Loui. *Theory and Computation of Uncertain Inference and Decision.* PhD thesis, University of Rochester Computer Science Department, September 1987.

[29] R. C. Nelson, D. H. Ballard, and S. D. Whitehead. Visual behavior and intelligent agents. In *SPIE Sensor Fusion II: Human and Machine Strategies*, volume 1198, November 1989.

[30] T. Olson and R. Potter. Real-time vergence control. In *Computer Vision and Pattern Recognition 1989*, June 1989.

[31] Richard N. Pelavin and James F. Allen. A model for concurrent actions having terporal extent. In *AAAI*, pages 246–250, 1988.

[32] M. L. Scott, T. J. LeBlanc, and B. D. Marsh. Design rationale for psyche, a general-purpose multiprocessor operating system. In *International Conference on Parallel Processing*, August 1988.

[33] Robert Fletcher Sproull. *Strategy Construction using a Synthesis of Heuristic and Decision-Theoretic Methods.* PhD thesis, Stanford University, Stanford, CA, 1977.

[34] Josh D. Tennenberg. *Abstraction in Planning.* PhD thesis, University of Rochester, Rochester, NY 14627, 1988.

[35] Josh D. Tennenburg and Jay C. Weber. A statistical solution to the qualification problem. Unpublished Manuscript, 1989.

[36] Steven Vere. Planning in time: Windows and durations for activities and goals. In *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5(3)*, pages 246–267, 1983.

[37] Jay C. Weber. A parallel algorithm for statistical belief refinement and its use in causal reasoning. In *Eleventh IJCAI,* August 1989.

[38] T. Winograd. A procedural model of language understanding. In Barbara J. Grosz, Karen Spark Jones, and Bonnie Lynn Webber, editors, *Readings in Natural Language Processing*, pages 249–266. Morgan Kaufmann Publishers, Inc., 1986.